COMPUTATIONAL THINKING

These ways of thinking help you to solve problems.

Aspect	Definition	Meaning	Theoretical positives / Negatives	Simple examples
Thinking abstractly	Removing unnecessary details and including only the relevant details.	Identifying what does and doesn't matter to solving the problem. The idea of layering or levels of a problem. Deciding what variables & objects will be needed.	+ Simplifies the problem / interface. + Less computation / data. + Easier to see how the solution to one problem can also be the solution to another. - Models will not be as accurate.	Icons / symbols on a map. Charting data. Moving nodes on a graph data structure to change how it looks.
Thinking ahead	Identifying the preconditions of a system: inputs, outputs, and reusable components.	What you need to know before you can solve the problem. The state of data for an algorithm to work. Identifying what data is required before it is needed (caching) Identifying reusable program components.	+ Caching can speed up a process. - Caching can be complicated to implement. - Caching requires the correct data to be fetched for the next instruction.	Working out how much paint you need before starting to decorate. Getting your debit card out before it is needed to be scanned. Sorting data for a binary search.
Thinking procedurally	Breaking a problem down.	Identifying several smaller sub-problems. Determine the order of events.	+ Problems are easier to solve. + Debugging is easier.	Generating a subject grade requires putting marks into a system, before applying a grade boundary, before printing results.

			- May not be entirely possible with an event driven rather than procedural approach to programming.	
Thinking logically	Identifying individual steps and decision points of an algorithm.	Identify the points at which a selection or iteration is needed. Determine the conditions of the decision. Determine the next steps depending on the outcome of the decision.	+ Makes writing an algorithm easier. + The complexity of an algorithm can be determined. + Algorithms can be simplified, or better solutions found more easily. + Identifies branches for testing.	Happens after thinking procedurally. Using a flowchart or pseudocode to identify the individual steps of an algorithm.
Thinking concurrently	More than one process happening at the same time.	Identifying parts of the problem can be executed at the same time.	+ Increase in speed. - May be difficult to program. - Can result in deadlock. - Problem may not suit concurrency.	When building a house, ordering the windows, while putting up the walls. Playing sound in a game while taking user inputs. Multiple images downloading for a webpage.